

ACTCCCACACCCCCTTTTCCCACACACCCTAAAACAGAACCCCACAGAGAATGTCCTCCCACC۲, ACCAGCATACCACCCGGCCTGTACCACGCCTGCCTGGCCTCGCTGTCAATCCTTGTCCTG eccedecute coecedecte coefficies and the constant of the coefficient of the coefficientCCTGTTTTCATCGTCCTCCTGAGCTCCCTGTCTTTGCTGCTCCCCGACGAGGACGCATTG 1.0 CCCTTC TGACTCTCGCCTCAGCACCCAGCCAAGATGGGAAAACTGAGGCTCCAAGAGGCC CCCACGGCFGCCCACACACCTGCACACCTGCTCGCAGCACCCTGTCCTGGGGCCCACCTT 15 TCCCFGCFGGCCTCCCTGCCFCCFGCTGGGGCCTCGGATTCCTGAGCCTTTGCTACCCT  $\operatorname{CTGCAGCTGGFGAGAGCTTFCAGCCGFACGACAGGAGCACGCTCCAAGGGGCTCCAAGAGCCC}$ ACCTACTCT/GACGAATATCTCACGAACCTCCTTTCCACGAACAAGCTGCGAAAGCAGCTAC CACACCTCCAACCATGGCTTCCTGTCCTCGGCCCGCGTCTGCTTGAGACACTGCATCTAC A CTCCAE AGCCCAGGATTCCATCTCCCGCTGAAGCTGGTGCTTTCAGCTACACTGACAGGG 20  $\operatorname{ctrgAccoc}_{Accoc}$ TCCGAG-ACAACCACGAGCTGGTGGAGCTGGTGAAGCAC ATCTGTGGGCTCTCGAAGTG TOUTACAGE TO AGE OF TEGET COTTOUT TACT CACCITY OF TEGET CUTGAT COGCITYA CTGGTGSCACACGACCATCCAGCTCTGCACCGACGAGGTGCCCTGGACTTGACT25  $cccrps_{CC} + \Delta T ccc \Delta G T C C C C \Delta T C C C C C C \Delta A G C C \Delta T \Delta T T C T G T T G A G C T T C A G T$  $cccep_{CC}(AdA) \land acccepte a to test consistent of the constant of the const$ CAAACCACOCCCTEGCCTTCCTEGCTGCTCATCCCTGTGCTCCATGGCAGGAACCTCCTG  ${\tt CTCFTCCCTTCCCTGGAGTCCTCGTGGCCTTCTGGCTGACTTTGGCCCTGGCTGATC}$ CTGCAGAACATGGCAGCCCATTGGGTCTTCCTGGAGACT/ATGATGGACACCCACAGCTG 3.0  $\texttt{ACCAAC} \in \texttt{GOCCAGTGCTATGCAGCCACCTTTCTTCTTCTTCTCCCCCTCAATGTCCTCGTG}$ ectroccapostoscoacerssedastscreerererereace refacaacoccafecacerr  ${\tt COCCAGATEGACCTCACCCTGCTGCCACCGAGAGCCGCCACTCTCGACCCCGGCTACTACTAC}$ A COTACC GAAACTTCTTGAAGATTGAAGTCAGCCAGTCGC ATCCAGCCATGACAGCCETC 35 AGCCTCAGACCAGGGGAGGAAGACGAAGGGATGCAGCTGCTACAGACAAAGGACTCCATG CCCAAGCCACCTACCCCCGCCCCAGCCCCCCCCAGGGCTCCCTGGCGTCTCGCCTACACG e Toottoe Acaa cocha coctoca obtetto cocaa da coccotto TOCOTTO COCAA TOOT 4 ()  $\operatorname{ccc}$   $\operatorname{ccc}$   $\operatorname{cc}$   $\operatorname{Acc}$   $\operatorname{$ CTGGGTTACCCCCTTGCTCCAGGAGCCAGTTGACCCAGGGCACCCACATCCAGGCGTCTC $\epsilon$  CTACC). TO COTOTOCCATCACCOTTGAAGGCCCTCGATGAAGCCTTCTCTGGAACCACT  $\tt CCAGCCCCAGCCTCAGCCTTGGCCTTCACCCTGTGGAAGCAGCCAAGGCACTTCCT$ ٠٦ t,  ${
m erg}{
m c}_{
m A}{
m d}{
m d}{
m c}_{
m A}{
m c}_{
m C}{
m A}{
m d}_{
m A}{
m c}_{
m A}{
m d}_{
m A}{
m c}_{
m C}{
m d}_{
m A}{
m c}_{
m c}_{
m A}{
m c}_{$ 

erzele elzer halls en mij aen help ala anzape elan in elektropythick AAACAC 1997-eigen ett ett 190

MCSQEACHQTSPGATEDTSYCSWYIDEPQCCEELQFEGEVPSCHTSIFICLTHACLASLS
LLVLELLAMEVFEQLWFDCVFGFFGLFSFVDFLACDFFFAVFAAVFMVLLSSLCLLLFD
EDALFFLTLASAPSQDGKTEAFEGAWKILGLFTYAALYYPLAACATAGHTAAHLEGSTLS
WAHLGVQVWQFAECFQVFKIYKYSLLASLFLLIGLGFESLWYFVQLVRSFSFRTGAGSK
GLQSSTSEEYLFDLLCFKKLGSSYHTSKHGFLSWARVCLFHCIYTPQFGFHLFLKEVLSA
TLTGTAIYQVALLLLVGVVFTIQKVRAGVTTDVSYLLAGFGIVLSEDKQEVVELVKHHLW
ALEVCYISALVLSCLLTFLVLMRSLVTHRTNLRALHRGAALDLSPLHRSPHFSRQAIFCW
MSFSAYQTAFICLGLLVQQIIFFLGTTALAFLVLMPVLHGRNLLFRSLESSWFFWLTLA
LAVILQHMAAHWVFLETHDGHPQLTDRRVLYAATFLLFPLNVLVGAMVATWRVLLSALYN
AIHLGQMDLSLLFPFRAATLDFGYYTYRNFLKIEVSQSHFAMTAFCSLLLQAQSLLFRTMA
APQDSLRFGEEDECMQLLQTKDSMAKGARFGASEGFARWGLAYTLLHDPTLQVFFKTALL
GANGAQP

Important features of the protein: Signal peptide:

None

 $1^{\circ}$ 

```
Transmembrane domain:
```

```
20
      4.69
      102 119
      148 166
      207 272
, t
      301 320
      364 380
      43.1 - 4^{3} \cdot 1
      474 489
      560 535
3 ()
      Motif file:
      Motif name: N glycosylation site.
             8 12
3 C,
      Motif name: N myristoylation site.
            1,0 56
           176 182
-1()
           241 247
```

the state of the s

Martin property Add SMD transfers a site of the entry to be a second or as

45

317 323 341 347 525 531 627 633

631 637 640 646 661 667

# FIGURE 3A

PRO

ι,

10

XXXXXXXXXXXXX

(Length = 15 amino acids)

Comparison Protein

XXXXXYYYYYYY

(Length = 12 amino acids)

% amino acid sequence identity -

(the number of identically matching amino acid residues between the two polypeptide sequences as determined by ALIGN-2) divided by (the total number of amino acid residues of the PRO polypeptide)

5 divided by 15 - 33.3%

# FIGURE 3B

PRO

Ι,

10

XXXXXXXXX

(Length : 10 amino acids)

Comparison Protein

XXXXXYYYYYYZZYZ

(Length == 15 amino acids)

% amino acid sequence identity

(the number of identically matching amino acid residues between the two polypeptide sequences as determined by ALIGN 2) divided by (the total number of amino acid residues of the PRO polypeptide)

5 divided by 10 - 50%

# FIGURE 3C

PRO DNA NNNNNNNNNNNNNNN (Length 14 nucleotides)

5 Comparison DNA NNNNNNLLLLLLLLL (Length 16 nucleotides)

% nucleic acid sequence identity =

(the number of identically matching nucleotides between the two nucleic acid sequences as determined by ALIGN 2) divided by (the total number of nucleotides of the PRO DNA nucleic acid sequence)

6 divided by 14 42.9%

11,

# FIGURE 3D

PRO DNA

ι,

10

ИИИИИИИИИИИИ

(Length = 12 nucleotides)

Comparison DNA

NNNNLLLVV

(Length 9 nucleotides)

% nucleic acid sequence identity =

(the number of identically matching nucleotides between the two nucleic acid sequences as determined by ALIGN 2) divided by (the total number of nucleotides of the PRO DNA nucleic acid sequence)

4 divided by 12 33 3%

### FIGURE 4A

```
* C. C. mereased from 12 to 15
        * 7 is average of EQ
        * B is average of ND
        * match with stop is M, stop stop = 0; F (joker) match = 0
        */
       #define
                 M
                          8
                                  /* value of a match with a stop */
1.0
       int
                 day[26][26]
               A B C D E F G H I J K L M N O P O R S F U V W X Y Z <sup>1</sup>/<sub>2</sub>
       /* A */
                  \{2, 0, 2, 0, 0, 4, 1, 1, 1, 0, 1, 2, 1, 0, M, 1, 0, 2, 1, 1, 0, 0, 6, 0, 3, 0\}.
       /* B */
                   {0, 3, 4, 3, 2, 5, 0, 1, 2, 0, 0, 3, 2, 2, M, 1, 1, 0, 0, 0, 0, 2, 5, 0, 3, 1},
11,
       / · C ·/
                   { 2, 4,15, 5, 5, 4, 3, 3, 2, 0, 5, 6, 5, 4, M, 3, 5, 4, 0, 2, 0, 2, 8, 0, 0, 5}.
       /* D */
                   \{0, 3, 5, 4, 3, 6, 1, 1, 2, 0, 0, 4, 3, 2, M, 1, 2, 1, 0, 0, 0, 2, 7, 0, 4, 2\}
                   \{0, 2, 5, 3, 4, 5, 0, 1, 2, 0, 0, 3, 2, 1, M, 1, 2, 1, 0, 0, 0, 2, 7, 0, 4, 3\},\
       /* E */
       /* F */
                   {4, 5, 4, 6, 5, 9, 5, 2, 1, 0, 5, 2, 0, 4, M, 5, 5, 4, 3, 3, 0, 1, 0, 0, 7, 5},
       /* G */
                   { 1, 0, 3, 1, 0, 5, 5, 2, 3, 0, 2, 4, 3, 0, M, 1, 1, 3, 1, 0, 0, 1, 7, 0, 5, 0},
20
       /* H */
                   { 1, 1, 3, 1, 1, 2, 2, 6, 2, 0, 0, 2, 2, 2, M, 0, 3, 2, 1, 1, 0, 2, 3, 0, 0, ?}.
       / * 1 */
                   { 1, 2, 2, 2, 2, 1, 3, 2, 5, 0, 2, 2, 2, 2, M, 2, 2, 2, 1, 0, 0, 4, 5, 0, 1, 2}.
       /* J */
                   /* K */
                   { 1, 0, 5, 0, 0, 5, 2, 0, 2, 0, 5, 3, 0, 1, M, 1, 1, 3, 0, 0, 0, 2, 3, 0, 4, 0}.
                  \{\,2,\,3,\,6,\,4,\,3,\,2,\,4,\,2,\,2,\,0,\,3,\,6,\,4,\,3,\,M,\,3,\,2,\,3,\,3,\,1,\,0,\,2,\,2,\,0,\,1,\,2\},
       /* L */
                   \{4,2,5,3,2,0,3,2,2,0,0,4,6,2,M,2,4,0,2,1,0,2,4,0,2,4\}
25,
       1 * 1.1 */
       /* N */
                   \{0, 2, 4, 2, 1, 4, 0, 2, 2, 0, 1, 3, 2, 2, M, 1, 1, 0, 1, 0, 0, 2, 4, 0, 2, 1\}
       1.0 1
                                                             0, M, M},
       /* P */
                  { 1, 1, 3, 1, 1, 5, 1, 0, 2, 0, 1, 3, 2, 1, M, 6, 0, 0, 1, 0, 0, 1, 6, 0, 5, 0},
3 ()
       /* Q */
                  {0, 1, 5, 2, 2, 5, 1, 3, 2, 0, 1, 2, 1, 1, M, 0, 4, 1, 1, 1, 0, 2, 5, 0, 4, 3}.
       /* R */
                  \{2, 0, 4, 1, 1, 4, 3, 2, 2, 0, 3, 3, 0, 0, M, 0, 1, 6, 0, 1, 0, 2, 2, 0, 4, 0\}.
       1 * 5 */
                  { 1, 0, 0, 0, 0, 3, 1, 1, 1, 0, 0, 3, 2, 1, M, 1, 1, 0, 2, 1, 0, 1, 2, 0, 3, 0},
       /* T */
                   \{1, 0, 2, 0, 0, 3, 0, 1, 0, 0, 0, 1, 1, 0, M, 0, 1, 1, 1, 3, 0, 0, 5, 0, 3, 0\}.
       /* U */
                  36,
       /* V */
                  \{0, 2, 2, 2, 2, 1, 1, 2, 4, 0, 2, 2, 2, 2, M, 1, 2, 2, 1, 0, 0, 4, 6, 0, 2, 2\}
       /* W */
                  { 6, 5, 8, 7, 7, 0, 7, 3, 5, 0, 3, 2, 4, 4, M, 6, 5, 2, 2, 5, 0, 6, 17, 0, 0, 6},
       /* X */
                  /* Y */
                  \{3,3,0,4,4,7,5,0,1,0,4,1,2,2,M,5,4,4,3,3,0,2,0,0,10,4\},
                  \{\, 0,\, 1,\, 5,\, 2,\, 3,\, 5,\, 0,\, 2,\, 2,\, 0,\, 0,\, 2,\, 1,\, 1,\, M,\, 0,\, 3,\, 0,\, 0,\, 0,\, 0,\, 2,\, 6,\, 0,\, 4,\, 4\}
       1 + 7. +1
40
       }:
```

4.

i, ()

٠,٠,

# FIGURE 4B

```
1.
         +/
        #include + stdio h +
        #include + ctype h +
        #define MAXJMP
                                     16
                                               /* max jumps m a diag */-
        #define MAXGAP
                                     24
                                               /* don't continue to penalize gaps larger than this */
        #define JMPS
                                      1024
                                               /* max imps in an path */
10
        #define MX
                                      . ‡
                                               /* save if there's at least MX I bases since last jimp */
        #define DMAT
                                      3
                                               /* value of matching bases */-
        #define DMIS
                                     0
                                               /* penalty for mismatched bases */
        #define DINSO
                                     8
                                               /* penalty for a gap */
15
        #define DINS1
                                     1
                                               /* penalty per base */
        #define PINSO
                                     8
                                               /* penalty for a gap */
        #define PINS1
                                               /* penalty per residue */
                                      1
        struct jump {
20
                                                         /* size of jmp (neg for dely) */
                  short
                                     n[MAXJMP];
                  unsigned short
                                      MMAXJMP],
                                                         /* base no of jmp in seq x */.
                                                         /* himits seq to 2°16 1.*/.
        }:
        struct diag {
25
                  int
                                     score,
                                                         7* scere at last jmp */.
                                     offict,
                                                         7* office of prev block */
                  long
                                                         /* current jmp mdex */
                  short
                                      ninp;
                                                         /* list of jmps */
                  struct imp
                                     IP;
        };
3.0
        struct path {
                                               /* number of leading spaces */
                  int
                            B[JMPS];/* size of imp (gap) */
                  short
                            x[JMPS]; /* loc of jmp (fast elem before gap) */
3 5
        }.
        char
                            *ofile;
                                                         /* output file name */
        char
                            *namex[2];
                                                         /* seq names, getseqs() */-
                                                         /* prog name for err msgs */
        char
                            'prog;
4.0
                                                         /* scqs. gctseqs() */
        char
                            *seqx[2];
                            dınax;
                                                         /* best diag: nw() */
        int
                                                         /* final drag */
        int
                            dmax0;
                                                         /* set if dual main() */
                            dna,
        int
                                                         /* set if penalizing end gaps */
        int
                            endgaps;
                                                         /* total gaps in seqs */
٠1 t,
        int
                            gapx, gapy;
                                                         7* seg lens */
                            len0, len1,
        int
                                                         /* total size of gaps */
        int
                            ngapx, ngapy,
        int
                            smax;
                                                         /* max score inv() */
                                                         /* bitmap for matching */-
        int
                            *xbin;
\mathbf{r}, \mathbf{0}
                                                         /* current offset in jinp file */
                           offset.
        long
        struct
                  diag
                            *dx;
                                                         /* holds diagonals */
                                                         /* bolds path for segs */
        struct
                  path
                           pp[2].
                            *callect), *imilloc(), *imfex(), *itrepyt),
        char
```

# FIGURE 4C

```
/* Needleman Wunsch alignment program
         * usage progs file1 file2
 f",
         * where file1 and file2 are two dna or two protein sequences
         * The sequences can be in upper or lower case an may contain ambiguity
         * Any lines beginning with 't', ' - ' or ' - ' are ignored.
         * Max file length is 65535 (himted by unsigned short x in the juip struct)
           A sequence with 1/3 or more of its elements ACGTU is assumed to be DNA
10
           Output is in the file "align out"
         * The program may create a tmp file in /tmp to hold into about traceback
         * Original version developed under BSD 4.3 on a vax 8650.
15
        #include "nw h"
        #include "day h"
        static
                 1,14,2,13,0,0,4,11,0,0,12,0,3,15,0,0,0,5,6,8,8,7,9,0,10,0
20
        };
                  phval[26] {
        static
                  1, 2[(1++(D^{*}/A))](1++(N^{*}/A)), 4, 8, 16, 32, 64,
                 128, 256, 0xHHHHHH, 1 < (10, 1) < (11, 1) < (12, 1) < (13, 1) < (14, 1)
25
                 1 < +15, 1 < +16, 1 < +17, 1 < +18, 1 < +19, 1 < +20, 1 < +21, 1 < +22
                 16 + 23, 16 + 24, 16 + 25[(16 + (16)(A)))](16 + CQ^{**}A^{*}))
        }:
                                                                                                                         main
        main(ac, av)
3 ()
                 int
                           *as []:
                 char
        1
                 brog
                         av[0];
                 if (ac †
                          3) {
31,
                           fprintf(stderr, "usage "%s file1 file2"n", prog);
                           fprintf(stderr, "where file1 and file2 are two dna or two protein sequences \n").
                           fprintf(stderr," The sequences can be in upper- or lower case'n"),
                           fprintf(stderr, "Any lines beginning with ',' or ' ' are ignored\n"),
                           fprintf(stderr, "Output is in the file \"align out\"\n").
40
                           exit(1);
                 }
                 namex[0]
                            av[1]:
                 namex[1]:
                            av[2].
                 seqx[0] = getseq(namex[0], \&len0),
41
                 seqx[1] getseq(namex[1], &fenD,
                 abin (dna)? dbval, pbval,
                                                       7* I to penalize endgaps 1/
                 endgaps 0,
                        "align_out";
                                                       /* output file */
                 ofile
1,()
                                    /* fill in the matrix, get the possible imps */
                 readimps();
                                    /* get the actual jumps */.
                                    /* print stats, alignment */
                 puntO,
```

# FIGURE 4D

```
/* do the alignment, return best score: main()
         * dna: values in Fitch and Smith, PNAS, 80, 1382-1386, 1983
         * pro: PAM 250 values
         * When scores are equal, we prefer mismatches to any gap, prefer
         * a new gap to extending an ongoing gap, and prefer a gap in seqx
         * to a gap in seq y
         • 1
                                                                                                                                 nw
        nw()
10
        {
                                      *px, *py;
                                                          7* segs and ptrs 37
                  char
                                                          /* keep track of dely */
                                      'ndely, 'dely,
                  int
                                                          /* keep track of delx */
                  int
                                      ndelx, delx;
                                                          /* for swapping row0, row1 */
                                      *unp;
                  int
10
                                                          /* score for each type */-
                  int
                                      mist
                                                          /* insertion penalties */
                  int
                                      ins0, ins1;
                                                          /* diagonal index */
                  register
                                      id;
                  register
                                                          /* prop index */
                                      ij,
                                                          /* score for curr, last row */
                                      *col0, *col1,
                  register
20
                                                          /* index into seqs */
                  register
                                      xx, yy;
                        (struct diag.*)g. calloc("to get diags", len0+len1+1, sizeof(struct diag.)).
                  dx
                           (int *)g_calloc("to get ndely", len1 +1, sizeof(int)),
25
                          (int *)g calloc("to get dely", lent +1, sizeof(int)),
                  dely
                         (int ')g_calloc("to get col0", len1 + 1, sizeof(int)),
                  col0
                         (int ')g calloc("to get col1", len1 + 1, sizeof(int)),
                  coll
                  ins0
                          (dna)? DINSO PINSO,
                         (dna)? DINST : PINST,
                  insl
3.0
                            I(XXX);
                  smax :
                  if (endgaps) {
                            for (col0[0] = dely[0] =
                                                     = ms0, yy = 1; yy \leftarrow -1en1; yy + + 1
                                     col0[yy] dely[yy] col0[yy]] insl,
şι,
                                      ndely[vv] vy.
                            cel0[0] 0;
                                                /* Waterman Bull Math Biol 84 */
                  }
                  else
                            for (yy - 1; yy \leftarrow = lcn1; yy++)
40
                                      delv[yy] = ins0;
                  /* fill in match matrix
45
                  for (px
                             seqx[0], xx = 1; xx \leftarrow len0, px + +, xx + +)
                            i* initialize first entry in cel-
                            if (endgaps) {
                                      if (xx
                                                 1)
(), ()
                                                ceH[0]
                                                           delx
                                                                   (in:0 tins1);
                                      else
                                                                  col0[0] and,
                                                col[[0]
                                                           delx
                                      ndelx
                                               X X .
                            3
```

}

# FIGURE 4E

 $\dots$ nw  $seqx[H, yy = 1; yy \leftarrow len1; py + +, xy + +)$ for (py mis col0[vv-1], 1, if (dna) mis 3 (xbm[\*px 'A']&xbm[\*py 'A']?/ DMAT: DMIS; else day[\*px 'A'][\*py 'A']. mis + 10 /\* update penalty for del in x seq. \* favor new del over ongong del-\* ignore MAXGAP if weighting endgaps if (endgaps || ndely[vv] < MAXGAP) { 15 if  $(colO[yy] + ms0 \rightarrow dely[yy])$  { delv[vv] = col0[vy] (ms0 + ms1); ndely[vy] = 1;} else { dely[yy] = insl;20 ndely[yy] + +. } } else { if  $(col0[vy] \cdot (ins0 + ins1) \rightarrow delv[yy])$  { delv[vv] = col0[vv] + (ms0 + ms1),  $ndels\{yy\}=1,$ 26 } else ndelv[vv] + +. } 3 () /\* update penalty for del in y seq; \* favor new del over ongong del if (endgaps | | ndelx + MAXGAP) { if (coll[yv]) imsO = delx) { З**г**, defx = coll[yy 1] - (ins0 + ins1);ndclx 1; } else { delx ms1; ndetx + +;40 } } else { if (coll[yy 1] - (ms0 + ins1) > - delx){  $dclx = coll[vy \ I] \cdot (ms0 \pm ms1),$ ndefx 1, ,11, } else ndelx++. } 7\* pick the maximum score; we're favoring r,() \* mis over any del and delx over dely

r r

## FIGURE 4F

...BW id = xx - yv + len1 - 1;if (mis - dely && mis - dely[vy]). coll[yy] mis; ٤, else if (dclx - dclv[vy]) {  $-\cos H[yy] = \det x;$ ij dx[id] jimp; if (dx[id] ip n[0] && ('dna | ] (ndelx - MAXIMP &&  $xx \rightarrow dx[id]$  ip x[ij] + MX) || mis  $\rightarrow dx[id]$  score + DINSO() { 10 dx{id} jimp { +; if  $(++ij) \rightarrow MAXJMP$ ) { writejmps(id);  $\eta = dx[id] ijmp = 0,$ dx[id] offset + offset;  $1^{\circ}$ offset + sizeof(struct jmp) + sizeof(offset); } dx[id] p[ij] = ndclx;dx[id] ip x[ij] = xx;20 defidiscore dela, } else {  $cotf{yy} = dety{yy}$ . 25 ij = dx[id].ijmp, if (dx[id] ip  $\mathfrak{n}[0]$  && ('dna  $\{[(\mathsf{ndely[yy]})\}^{-1}\}$  MAXIMP. &&  $\chi\chi \to d\chi[id]$  )p  $\chi[ii] + MX)$  [] mis  $\to d\chi[id]$  score + DINSO)) {  $\frac{dx[id]|\eta mp + +,}{if(++\eta) = -MAXJMP}$ 3 () wintejmps(id), ij dx[id] jjimp 0, dybdl offset - offset, offset + sizeof(struct pup) + sizeof(offset); ζ£, ndclv[yy], dx[id] ip n[ii]. dx[id](p,x[n) = xx;dx[id] score = dcly[yy]; 4() if (xx len0 && yy ← len1) { /\* last col-+/ if (endgaps) coll[yy] ins0 + ins1\*(len1 yy); 40 if (coll[vv] > smax) { smax = coll[yy].id; dmax τ, () } if (endgaps && xx < fen0). coll[vy 1] = ms0 + ins1\*(len0 vo);if (coll[yy I] - smax) { max coll[vv I].  $\mathbf{r}_{i}$ 

(void) free(char \*) dely);
(void) free(char \*)dely);
(void) free(char \*)col0);(void) free(char \*)col1);}

Page 4 of nw c

## FIGURE 4G

```
* print()
                  only reutine viable outside this module
 ι,
         * static:
         * getmat() - trace back best path, count matches: print()
         * pr. align() - print alignment of described in array p[]. print()
         * dumpblock() - dump a block of lines with numbers, stars: pr_align()
10
         * nums() put out a number line: dumpblock()
         * puthoc() - put out a line (name, [num], seq. [num]); dumpblock()
         * stars() put a line of stars: dumpblock()
         * stripname() -- strip any path and prefix from a sequame
11,
        #include "nw h"
        #define SPC
        #define P LINE 256
                                    /* maximum output line */
2.0
        #define P_SPC
                           3
                                    7* space between name or num and seq */
        extern
                  day[26][26];
        int
                 olen;
                                     /* set output line length */-
        \mathbf{H}\mathbf{H}\mathbf{E}
                  fx;
                                    /* output file */
25
                                                                                                                          print
        print()
        {
                                                      /* overlap */
                 int
                           Ix, Iy, firstgap, lastgap;
3 ()
                 if ((fx
                           fopen(ofile, "w"))
                                                  0) {
                           fprmif(siderr, "%'s can't write %s\n", prog. ofile).
                           cleanup(1),
                 fprintf(fx, "< first sequence: %s (length = %d)\n", namex[0], len0);
35
                 fprintf(fx, "< second sequence: %s (length = %d)\n", namex[1], len1);
                 elen 60,
                 lx = len0;
                 ly - len1;
                 firstgap = lastgap = 0;
40
                 if (dmax < len1 - 1) {
                                            /* leading gap in x */
                          pp{0} spc = firstgap : len1 | dmax - 1;
                           ly == pp[0] spc;
                 else if (dmax - lcnl - 1) { /* lcading gap in y */
45
                           pp[1] spc firstgap dmax (len1 1);
                           lx pp[l] pc.
                 if (dmax0 < 4cn0 - 1) {
                                              7* trading gap in x */
                          Tastgap TenO dmaxO T;
t, ()
                          Ix - lastgap;
                 else if (dmax0 > lcn0 - 1) { /* trailing gap in y */
                           lastgap : dmax0 (lcn0 - 1);
                           Iy lastgap,
i r
                 1
```

# FIGURE 4H

```
* trace back the best path, count matches
         •/
 \Gamma_{5}
        static
                                                                                                                              getmat
        getmat(fx, fy, firstgap, lastgap)
                                                           /* "core" (minus endgaps) */
                  int
                            Ex, ly;
                                                           /* leading trailing overlap */
                            firstgap, lastgap;
                                      nm, i0, i1, siz0, siz1;
10
                  int
                                      outx[32];
                  char
                                      pct;
                  double
                                      n0, n1;
                  register
                                       *p0, *p1;
                  register char
1 \stackrel{\Box}{\rightarrow}
                  /* get total matches, score
                   •/
                   j0 \sim j1 \sim siz0 = siz1 \simeq 0;
                   p0 = seqx[0] + pp[1].spc;
                   p1 = seqx[1] + pp[0] spc;
20
                         pp[1] spc + 1;
                         pp[0].spc + 1;
                        =0;
                   nın
                   while ( *p0 && *p1 ) {
25
                             if ($i20) {
                                       pl++,
                                       nI + +1
                                       \sin 0 \sim 1
 3.0
                             else if (siz1) {
                                       p0++;
                                        n() + +;
                                       sizI--;
 35
                             }
                             else {
                                        if (xbm[*p0-'A']&xbm[*p1-'A'])
                                                  nm t t;
                                        if (n0 + + ... = pp\{0\}, x[i0])
                                                  siz\theta = pp[\theta].n[i\theta + +];
 40
                                        if (n1++ = - pp[1].x[i1])
                                                  siz1 = pp[1].n[i1 + +];
                                        p0 + +;
                                        p1++;
 .1 .
                              }
                    /* pct homology:
                     * if penalizing endgaps, base is the shorter seq.
                     * else, knock off overhangs and take shorter core
 €,()
                    if (endgaps)
                                    (len0 + len1)? len0 : len1;
                              ŀχ
                    else
                                    dx - Iv)" Ix Iv:
 , ,
```

. . .

```
...getmat
                  fprintf(fx, " + gaps in first sequence: %d", gapx),
                  if (gap v) {
                            (void) sprintf(outx, " (%d %s%s)".
 ٤,
                                                                                    1)? ""."s");
                                     ngapx, (dna)? "base"; "residue", (ngapx
                            formit(fx, "%s", outx);
                  fprintf(fx, ", gaps in second sequence: %d", gapy);
10
                  if (gapy) {
                            (void) sprintf(outx, " (%d %s%s)",
                                      ngapy, (dna)? "base": "residue", (ngapy
                                                                                     1)? "":"s");
                            fprintf(fx, "%s", outx);
                  if (dna)
\mathbf{l}^{-1}
                             fprintf(fx.
                                                                          %d, gap penalty — %d + %d per base)\n",
                             "In e score: %d (match = %d, mismatch
                            smax, DMAT, DMIS, DINSO, DINSO;
                   else
                             fprintf(fx,
20
                             "\n < scote %d (Dayhoff PAM 250 matrix, gap penalty — %d + %d per residue)\n",
                             smax, PINSO, PINS1);
                   if (endgaps)
                             fprintf(fx,
                             ** endgaps penalized, left endgap. %d %s%s, right endgap: %d %s%s\n*,
25
                             firstgap, (dna)? "base" = "residue", (firstgap = 1)? "" = "s", lastgap, (dna)? "base" = "residue", (lastgap = 1)? "" = "s");
                   else
                             fprintf(fx, " < endgaps not penalized\n");
 3.0
         }
                                                 /* matches in core -- for checking */
                             nm:
          static
                                                 /* lengths of stripped file names */
          static
                             lmax;
                                                 /* jmp index for a path */
                             ii[2];
          static
                                                 /* number at start of current line */
 35
          static
                             nc[2]:
                                                 /* current elem number - for gapping */
                             m[2]:
          static
          static
                             siz[2];
                                                 /* ptr to current element */
          static char
                              *ps[2];
                                                 /* ptr to next output char slot */
                              *po[2];
          static char
                                                /* output line */
          static char
                             out[2][P 1 INE];
 40
                                                 /* set by stars() */
                             star[P_LINE];
          static char
           * print alignment of described in struct path pp[]
 15
          static
                                                                                                                            pr align
          pr align()
          {
                                                  /* char count */
                                        nn;
                    int
                                        merc;
 r, ()
                    int
                    register
                                        i,
                              0, \text{Imax} = 0; i < 2; i + 3)
                    for (i
                              nn stripname(namex[i]);
                              of one lines
                              11/41
                                       H(1) ().
  + - (+
                               p^{*}[t]
                                       હવુપીતી.
                                                                                                     Page 3 of nyprint c
                               polit
                                       out il.
```

# FIGURE 4J

```
...pr align
                 for (m mn 0, more 1; more, ) {
                                   more 0; i \in 2, i \in \mathbb{N}
                           for (i
 ε,
                                     • do we have more of this sequence?
                                     if (!*ps[i])
                                              continue;
10
                                     more + +;
                                     if (pp[i].spc) { /* leading space */
                                               *po[i] \pm \pm \sim 1.7
                                               pp[i].spc -;
15
                                     }
                                     else if (siz[i]) { /* in a gap */
                                               ·pohl++
                                               siz[i] 💠
                                      }
20
                                                         /* we're putting a seq element
                                      else {
                                                *po[i] *ps[i];
                                                if (islower(*ps[i]))
                                                        - *ps[i] = toupper(*ps[i]);
25
                                                po[i] + + i
                                                ps[i] + +;
                                                * are we at next gap for this seq?
 3 O
                                                .,
                                                if (m[i] = \sup_{f \in \mathcal{F}} pp[i].x[i][i]) {
                                                          * we need to merge all gaps
                                                          * at this location
 35
                                                          +/
                                                          siz[i] = pp[i] \; n[i][i] + + \};
                                                          while (m[i] = = pp[i].x[ij[i]])
                                                                   siz[i] + z pp[i].n[i][i] + +];
 40
                                                }
                                                m[i] + + ;
                                      }
                             if (++nn : - ofen | | !more && nn) {
                                      dumpblock();
 10
                                      for (i = 0, i \leftarrow 2; i + 1)
                                              po[i] out[i];
                                      nn O;
                             }
 1,()
                   }
          }
           * dump a block of lines, including numbers, stars: pr_align()
 t 1
```

for (i = 0, i + 2; i f f)  $\label{eq:formula} \{ p(\mathbf{i}) = \frac{2\pi i}{2\pi} \}$ 

Page 4 of awprint c

# FIGURE 4K

...dumpblock (void) pote('\n', fx); ۲, for (i = 0, i < 2; i + 1)if (\*outh) && (\*outh) \* \* \* \* | \*(polit) \* \* \* \*) { if (ı (1) nums(r), if (i 0 && \*out[1]) 10 stars(); putlinc(i); 0 && \*out[]]) if (ı fprintf(fx, star);if (i 1)  $l^{\tau_i}$ noms(i); } } } 20 \* put out a number line; dumpblock() static nums nums(rx)  $2^{\mathbf{r}_i}$ /\* mdex in out[] holding seq line \*/ nline[P 11NF]; char register 1, }; \*pn, \*px, \*py; register char 3() for  $(pn = nline, i = 0, i < linex+P_SPC, i++, pn++)$ \*pn  $nc[ix], py = out[ix]; *py; py + +, pn + +) {$ for (i •рв • ру if (\*py =3 L, else { if (13710) 0 || (i - 1 && nc[ix]! 1)) {  $\mathbf{j} = (\mathbf{i} + 0)? \ \mathbf{i} : \mathbf{i};$ for (px = pn; j; j / 10, px-) \*px j%10 + '0'; 40 if (i < 0)\*px : ': } else \*pn ' '; .11, 111; } .70. \*pn 1,() nc[ix] i; for (pn = nline; \*pn; pn + +) (void) putc(\*pn, fx); (void) putc('\n', fx), } putline portline (ix) mt ix.

Page 5 of awprint c

# FIGURE 4L

```
...putline
                int
 ι,
                register char
                                   *px;
                         namex[ix], i = 0, 'px && 'px ! '''; px + +, i + +)
                         (void) putc(*px, fx);
                for (; i inax + P SPC, i + +)
                         (void) putc(' ', fx);
10
                /* these count from 1:
                 * m[] is current element (from 1)
                 * nc[] is number at start of current line
11,
                for (px = out[ix]; *px; px + ±)
                         (void) purc(*px&0x7F, fx);
                (void) putc('\n', fx);
       }
20
        * put a line of stars (seqs always in out[0], out[1]); dumpblock()
25
       static
                                                                                                                    stars
       stars()
       {
                                   *p0, *p1, cx, *px.
                 register char
3 ()
                                                              ...) ]]
                                        if (**out[0] || (*out[0] -
                   !*ent[]] [] (*out[])
                         return;
                 px - star;
şτ,
                 for (i = lmax + P SPC; i; i )
                          *px + +
                 for (p0 = out[0], p1 = out[1]; *p0 && *p1; p0++, p1++) {
                          if (isalpha(*p0) && isalpha(*p1)) {
-10
                                   if (xbm[*p0-'A']&xbm[*p1-'A']) {
                                            CX = '*';
                                            nm + +;
                                   }
                                   else if ('dna && | day['p0 'A']['p1 'A'] > 0)
.11,
                                            (x '';
                                   {\it else}
                                            ex · · · ,
                          }
1,()
                          else
                          *px++ cx;
                 }
'px++ "n'i
```

# FIGURE 4M

	* strip path or prefix from pn. return lenc pr. alignO */								
ι,	static stripnam	c(pn) Char	*pn;	/* file name (may be path)	•/		stripname		
10	{	register  py 0;  for (px	рв; <b>*</b> рх	'px, 'py; px++)					
15		if (py)	if (*px	7') py - px + 1; cpy(pn, py);					
20	}								
25									
30									
35									
40									
-1 r,									
τ,()									

1.5

# FIGURE 4N

```
7.
         * cleanup() - Cleanup any trop file
          * petseq() = read in seq, set dua, len, maxlen
  ι,
          * g_caffoc() = calloc() with error checkin-
          * readjinps() get the good jinps, from top file if necessary
          * write;mps() - write a filled array of jmps to a tmp file. nw()
         #include "nw b"
} ()
         #include + sys/file h +
                   *jname "/timp/homgXXXXXX";
                                                                     - /* tmp file for jmps */
         char
         HIE
                    * f1.
\Gamma^{\tau},
         int
                   cleanup();
                                                                      /* cleanup trip file */
                   Pæck⊕,
         long
         7.
          * remove any trip file if we blow
20
                                                                                                                             cleanup
         cleanup(i)
                   int
         {
                   if (fj)
25
                             (void) unlink(jname);
                   exit(i),
         }
 30
         * read, return ptr to seq, set dua, len, maxlen
          * skip lines starting with \langle \zeta_{i}^{\prime}\rangle \leq \zeta_{i} or \zeta_{i}^{\prime} \leq \zeta_{i}
          * seq in upper or lower case
         char
 31,
                                                                                                                               getseq
         getseq(file, Jen)
                             file,
                                       /* file name */
                   char
                                       /* seq len */
                              Hen;
                   int
         {
                                       line[1024], *pseq;
                   char
4 ()
                   register char
                                        *px, *py;
                   int
                                       natgo, tlen;
                   FILE
                                        * fp;
                   if ((fp.
                             fopen(file,"r"))
                                                 0) {
4.
                             tprintf(stderr, "%s: can't read %s\n", prog. file);
                             exit(D),
                   }
                   tlen nated 0,
                   while (fgcts(lmc, 1024, fp)) {
r,()
                             if (*Inc
                                       Time
                                                             ' | | 'line
                                       continue;
                                       bne; *px! = '\n'; px++)
                             for (px
                                       if (isapper('px) | islower('px))
                                                 tion ( )
```

```
...getseq
                 py pseq 1-4.
Hen tlen,
 ٢,
                 rewind(fp),
                 while (fycts(fmc, 1024, fp)) {
                                                        ' · ' || 'Ime
                                     ∵ || ¹line
                           if (*line
                                    continue;
                                    Inc, px! = \ln px + 1
10
                           for (px
                                    if (isupper('px))
                                              *py + +
                                    else if (istower(*px))
                                              *py + + toupper(*px);
                                    if (index("ATGCU",*(pv I)))
11,
                                              natge + +;
                           }
                  *py + + _ '\0',
                  *py - '\0';
20
                 (void) fclosc(tp);
                 dna i natgo i (den/3),
                 return(pscq +4),
25
        char
                                                                                                                     g calloc
        g_calloc(msg, nx, sz)
                           ^{*}m^{\circ}g_{*}
                                              /* program, calling routine */-
                 char
                                              /* number and size of elements */
                           nx, 52;
3 ()
                                     *px, *callect);
                 char
                                                                    (1) {
                 if ((px
                           calloc((unsigned)nx, (unsigned)sz))
                           if (*msg) {
                                    fprintf(stderr, "%s: g_calloc() failed %s (n - %d, sz - %d)\n", prog, msg, nx, sz);
31,
                 return(px);
40
         * get final jmps from dx[] or tmp file, set pp[], reset dmax; main()
                                                                                                                   readjmps
        readimps()
41,
        1
                                     siz, iO, iI.
                  register i, j. xx;
f , ( )
                  if (fj) {
                           (void) fclosc(fj);
                           if ((fd = open(mame, O_RDONLY, 0)) + = 0) {
                                     fprintfestderr, "Ts can't open() %s'n", prog. mame);
                           schile er 🕌
4 4 4
                                    for G = dyldmay prop 1.
                                                                 - 10 & & do [dma+] ip 対社
                                                                                              Page 2 of nwsubr.c
```

### FIGURE 4P

...readjmps

```
if (j + 0 && dyldmax) offset && f) {
                                                                                                                          (void) beck(fd, dx[dmax] offset, 0);
                                                                                                                          (void) read(fd, (char *)&dx[dmax] ip, sizeof(struct impi);
    ι,
                                                                                                                          (void) read(fd, (char *)&dx[dmax] off ct, sizeof(dx[dmax] off ct)),
                                                                                                                          dx[dinax] jimp MAXJMP 1;
                                                                                                  }
                                                                                                 else
                                                                                                                          break;
10
                                                                         }
if (i ->
                                                                                                 JMPS) {
                                                                                                  fprmtf(stderr, "%s: too many gaps in alignment\n", prog);
                                                                                                 cleanup(1);
                                                                         }
if (j
 \Gamma_{\Gamma_{2}}
                                                                                                 (1) {
                                                                                                 SIZ
                                                                                                               dx[dmax] ip n[i];
                                                                                                  xx = dx[dmax] ip x[i].
                                                                                                 dmax + siz,
                                                                                                                                                                             7* gap in second seq */
20
                                                                                                  if (siz < 0) {
                                                                                                                          բթ[1] ո[ւ1]
                                                                                                                                                                  517;
                                                                                                                           XX + S17,
                                                                                                                           /*id xx yy + lenl I
                                                                                                                            +/
24,
                                                                                                                          pp[1] \times [n] = xx \cdot dmax + lent = 1,
                                                                                                                           papy + + .
                                                                                                                           ngapy to.
                      /* ignore MAXGAP when doing endgaps */-
                                                                                                                           siz (siz - MAXGAP]} endgaps)? siz : MAXGAP;
  3()
                                                                                                                           11 + +,
                                                                                                  else if (siz > 0) { /* gap in first seq */
                                                                                                                           pp[0] n[10] siz.
  ξ.
                                                                                                                           pp[0] \times [0] = xx,
                                                                                                                           papx + +;
                                                                                                                           ngapx 1 Siz;
                       /* ignore MAXGAP when doing endgaps */
                                                                                                                           siz (siz + MAXGAP [] endgaps)? siz : MAXGAP,
40
                                                                                                                           jO + + 0
                                                                                                  }
                                                                         }
                                                                         else
                                                                                                  break,
41.
                                               }
                                                /* reverse the order of jmps
                                                                     \{0, [i0:]; j \leftarrow [i0:] j \pm \mp, [i0:] \}
                                                for (j
                                                                         i - pp[0] n[i]; pp[0] n[i] - pp[0] n[i0]; pp[0] n[i0] -
 t,()
                                                                         r = pp[0].x[i]; pp[0].x[i] = pp[0].x[i0], pp[0].x[i0] =
                                                for (i
                                                                     (0, i1- ; j < 11, j + 1, i1 ) {
                                                                         i pp[1] n[i], pp[1] n[i] = pp[1] n[i], pp[1] n[i] = i, i = pp[1] x[i] 
 t=t^{-},\\
```

evoids unitask pasies Transport

offset  $\{0,\}\}$ 

Page 3 of nwsubric

# FIGURE 4Q

```
* write a filled jmp struct offset of the previous (if any); inw()
                                                                                                                    writejmps
        writejmps(ix)
                 int
                           130
        {
                           *mktemp();
                 char
10
                 if (!fi) {
                           if (mktemp(jname) < 0) {
                                     fprintf(stderr, "%s can't niktemp() %s'n", prog. jname);
                                     cleamp(D);
1 5
                           if (cf)
                                    fopen(jname, "w")) 0) {
                                     fprintf(stderr, ""es can't write "es\n", prog. jname);
                                     exit(1),
                           }
20
                  (void) fwrite((char *)&dx[ix] jp, sizeof(struct jmp), 1, fj);
                  (void) fwrite((char *)&dx[ix] offset, sizeof(dx[ix] offset), 1, fp);
        }
25
3 ()
şг,
-1()
A^{-1} +
٠, ()
```

. .

 $\phi_{A}\phi_{A}$   $\phi_{A}\phi_{C}$   $\phi_{A}\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{A}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}\phi_{C}$   $\phi_{C}\phi_{C}\phi_{C}$ eccepter precedence and eccepter precent of the decay o١, CAAAACCAAACACACAACCAAGCGAGAGACGTCAGGACAGCGCCTTGATTGGAGGAGAAGGGGCC AGAGA**ATG**TEGTEGEAGCCAGCAGGGAACCAGACCPCCCCGGGGGCCACACAGAGACTACT TGTCAATCCTTGTGCTGCTGCTCCTTGCCCATGCTGAGGCGCCGCCAGCTCTGGCCTG 10 ACTGTGTGCGTGGCAGGCCCGGCCTGCCCAGGCCCCGGGCAGTGCCTGCTGCTGTTTTCA TCCTCCTCCTGAGCTCCCTGTGTTTGCTCCCCCGACGAGGACGCATTGCCCTTCCTGA CTCTCCCCTCAGCCAGCCAAGATCGGAAAACTGAGGCTCCAAGAGGGGCCTGGAAGATACTGCGACTGTTCTATTATGCTGCCCTCTACTACCCCTCTGGCTGCCTGTGCCACGCCTG GCCACACACCACCTGCTACACCTGCTCGGCAGCACGCTCCTGGGCCCACCTTGGCGGTCCAGG $\Gamma$ TCTGGCAGAGGGCAGAGTGTCCCCAGGTGCCCAAGATCTACAAGTACTACTCCCTGCTGG COTCCCTGCCTCTCCTGCTGGGCCTCGGATTCCTGAGCCTTTGGTACCCTGTGCAGCTGG TCAGAAGCTTCAGCCGTAGGACAGGAGCAGGCTCCAAGGGGGCTGCAGAGCAGCAGCTACTCTG AGGAATATCTGAGGAACCTCCTTTGCAGGAAGAAGCTGGGGAAGCAGCTACCACACCTCCA AGCATGGCTTCCTGTCCTGGGCCCGCCTCTCCTTCAGACACTGCATCTACACTCCACAGC 20 CAGGATTCCATCTCCCGCTGAAGCTCGTGCTTTCACCTACACTGACAGGGACGGCCATTT ACCAGGGCCCTGCTGCTGCTGGCCGGTGGTACCCACTATCCAGAAGGTGAGGGCAGAGCACGACCTCCTCCACCTCCAACCAACCACCATCTCTCGCCCTCTCCAACTCTCCTACATCT CAGGCTTGGTGTTGTCCTGCCTTACTCACCTTCCTGGTCCTGATGCGCTCACTGGTGACAC 25 ACAGGACCAACCTTCCAGCTCTCCACCCAGCAGCTCCCCTGGACTTGAGTCCCTTGCATC cometree contracted contracted and $c_{AG} c_{C} c_{T} T A T c_{T} c_{C} c_{T} T c_{G} c_{G} c_{T} c_{G} c_{G} c_{A} c_{A} c_{A} c_{A} c_{A} T c_{T} T c_{T} T c_{G} c_{A} c_{A} c_{A} c_{A} c_{G} c_{G} c_{A} c$ CCCTGGCCTTCCTGGTGCTCATGCCTGTCCATGGCAGGAACCTCCTGCTCTTCCGTTCCCTGCAGTCCTCGTGGCCCTTCTGGCTGACTTTGGCCCTTGGCTGTGATCCTGCAGAACA3 () TGGCAGCCCATTGGGTCTTCCTCGAGACTCATGATGGACACCCACAGCTGACCAACCGGC  $e_{A}$  is the content of the transfer of the content of the cont  ${\tt TGGGCACCTGGCGAGTGCTCCTCTGCCCCTCTACAACGCCATCCACCTTGGCCAGATGG}$ ACCTCAGCCTGCTGCCACCGAGAGCCGCCACTCTCGACCCCGGCTACTACACGTACCGAA ACTTCTTGAAGATTGAAGTCAGCCAGTCGCATCCAGCCATGACAGCCTTCTGCTCCCTGC 35  $\verb"TCCTGCAAGCGCAGAGCCTCCTACCCAGGACCATGGCAGCCCCCAGGGACAGCCTCAGAC$ CAGGGGAGGAAGAGGAAGGGATGCAGCTGCTACAGACAAAGGACTCCATGGCCAAGGGAG  ${\tt CTAGGCCCCGGGCAGCCGGCAGGGGCTCCCTGGGGTTTGGCTACACGCTGCTGCACAC}$ ACCCAACCCTCCAGGTCTTCCCCAAGACGGCCCTGTTGGGTGCCAATGGTGCCAGCCC ${f T}$ 40 crect procedure coese contrar correct Associate Active Associated Article Association and the contrar contrar and the contrared association and the contreGGATCACTGTGGTTGGGTGGAGGTCTCTTGCACTGGGAGCCTCAGGAGGGGCTCTGCTCC corrected aggreent aggreentTCTGCCATCAGCCTTGAAGGGCCTTCGATGAAGCCTTCTCTGGAACCACTCCAGCCCAGCT $4^{\text{t}}$  $ce \underline{\mathtt{A}} ce \underline{\mathtt{T}} e \underline{\mathtt{A}} ce e\underline{\mathtt{T}} \underline{\mathtt{T}} e \underline{\mathtt{A}} ce \underline{\mathtt{C}} \underline{\mathtt{T}} \underline{\mathtt{C}} \underline{\mathtt{T}} \underline{\mathtt{C}} \underline{\mathtt{A}} \underline{\mathtt{A}} \underline{\mathtt{C}} \underline{\mathtt{A}} \underline{\mathtt{A}} \underline{\mathtt{C}} \underline{\mathtt{A}} \underline{\mathtt{A}} \underline{\mathtt{C}} \underline{\mathtt{A}} \underline{\mathtt$ 

is the comparison which is the magnetic field of the federal  $\mathcal{D}(G)$ 

MSSQPAGNOTSEGATEDYSYGSWYTDEFQGGEELQFEGEVESCHTSTEPGLYHACLASL STEVELLELAMEVERROEWEDCVEGPEGLERERAVEAAVEMVELSSECTELFDEDALFEE TLACAPOPORTEAPROAWKILGLEYYAALYYPLAACATAGHTAAHLLGSTLSWAHLGV OVWORAECTOVPKTYKYYSILASIPLILGIGFISLWYPVOLVRSFSFRTGAGSKGLOSS YSEEYLRILLCRKKLGSSYHTSKHGFLSWARVCLRHCIYTFOFGFHLFLKLVESATLTG TATYOVALLLLVGVVPTTOKVRAGVTTDVSYLLAGFGIVLSEDKOEVVELVEHHLWALE VCYTSALVESCLETFEVEMRSEVTHRTNERALHRGAALDESPEHRSPHPSRQATFCWMS FSAYOTAFICLGLLVQQIIFFLGTTALAFLVLMPVLHGRNLLLFRSLESSWPFWLTLAL 10 AVILONMAAHWVELETHOGHEQLTNERVLYAATELLEELNVLVGAIVATWRVLLSALYN ATHLGOMDESTEPPRAATEDFGYYTYRNFEKTEVSOSHPAMTAFCSELLOAOSTEPRTM AAFORSLREGEEDEGMOLLOTKDSMAKGAREGASEGRARWGLAYTLLHIIFTLOVERKTA LLGANGAOP. 15 Important features of the protein: Signal peptide: none 2.0 Transmembrane domain: 54 71 93 111

140 - 157

25 197 214

3 ()

36

40

1.1

291 312

356 371

425 444

- 464 481 - 505 522

Motif name: N glycosylation site.

8 12

Motif name: N myristoylation site.

 $r_{i}(t) = r_{i}(t)$ 

167 173

137 138

308 3**14** 

332 338

516 522 618 624

45 622 628

631 637

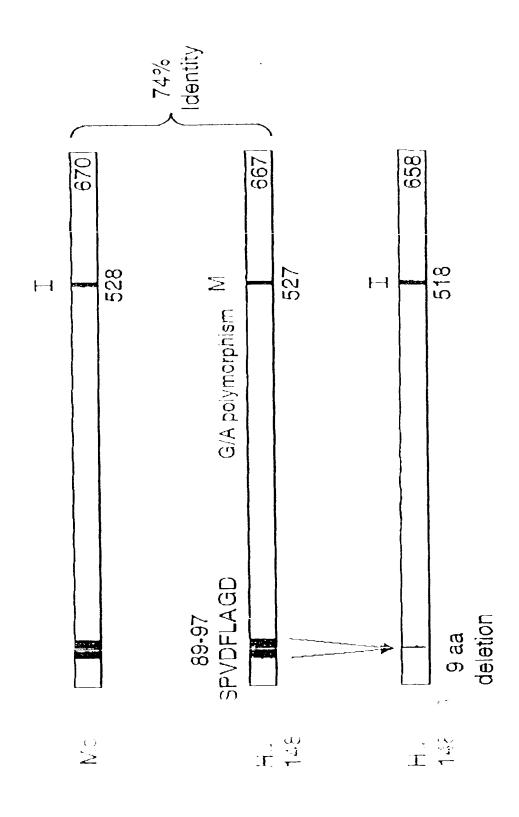
 $e_{\mathbf{r}}$ ,  $e_{\mathbf{r}}$ 8

rtain names to karyetic membrane lipeprotein lipid attachment

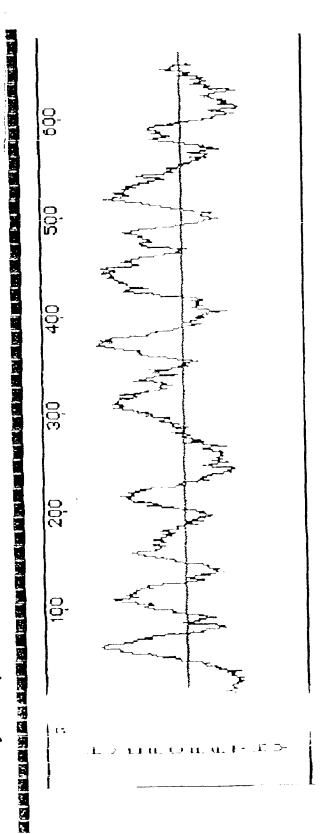
Hotel power offices building a to reduce a temperature

1,57 1.61

# Straß Variant Clones

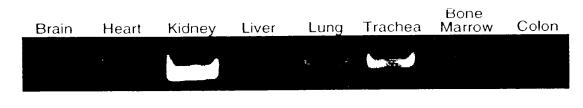


# Hyer aphobicity Plot of Human Stra6



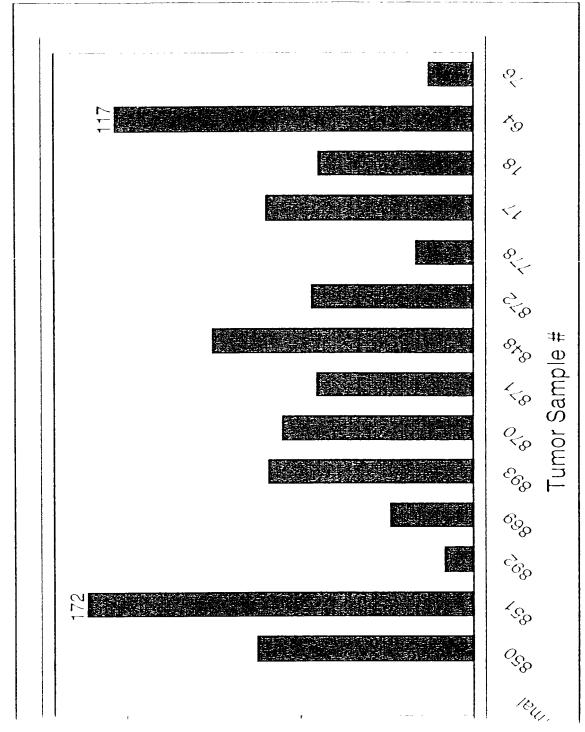
367 Amino Acids -->50% Residues Hydrophobic 73.5 kDa Protein 3 kb mRNA

3 Potential Transmembrane Domains



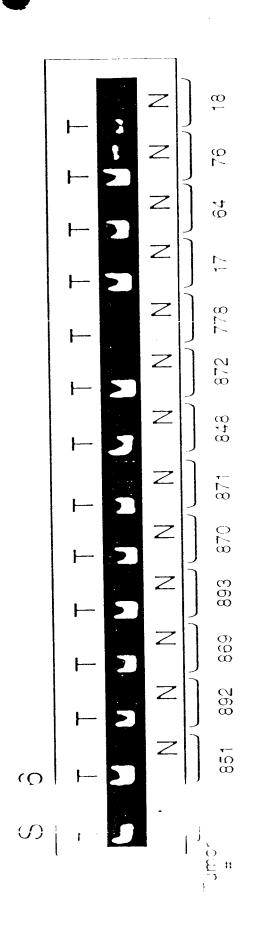
Breast Spleen Stomach Thymus Intestine Prostate Skeletal Muscle Testis Uterus

RNA Expression in Human Colon Tumor Tissue



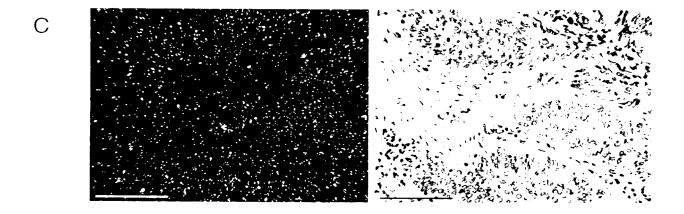
# ue vs Normal Mucosa From the Same Patient 3 RNA Expression in Human Colon Tumor

Taqman Product Analysis After 40 Cyoles



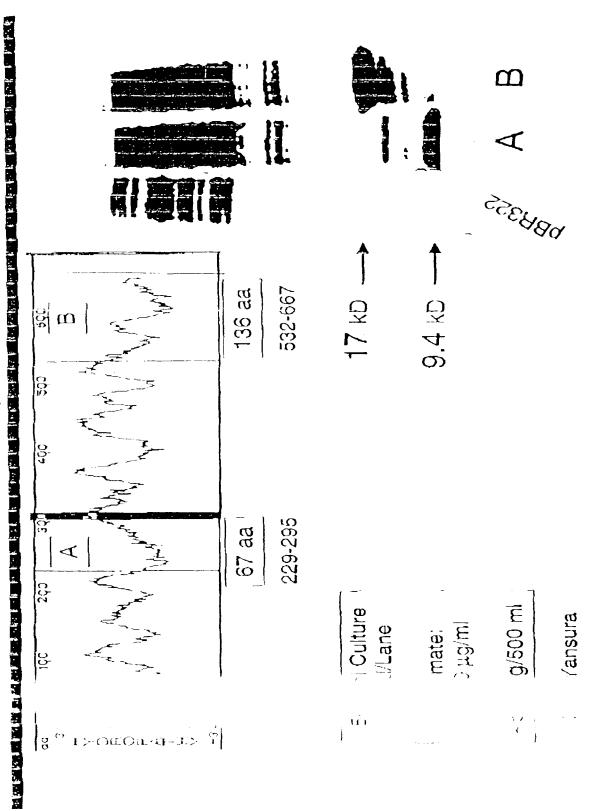
古

### FIGURE 12B



р С Э	Charles Charle
Colon	To the state of th
Kidney	
ast	

Poly-His Cleavable Leader at N-Terminus Peptide Expression in E. coli 



# Stra6 RNA Expression in Human Colon Carcinoma Cells +/- Retinoic Acid

TM#75 (2/28/00)

VD3 - vtamin D3 (TuM), ATRA - ali-trans-retinolo acid it uM) 90RA - 9-cis-retinolo acid it uM)

DAISO VOS ATER 8000 9000 14.09 4.09 4.09 + 67.FT 多数数 1

Cell Line / Treatment

4 (E. }-

04/180

+ 9cmA

+ VD3

4

